REMARKS

Applicant is in receipt of the Office Action mailed April 20, 2005. Claims 1, 29, 30, 45, 46, 56, 65, and 73 have been amended. Claims 1-2, 5-33, 36-47, 50-59, 62-65, 68-76, and 79-81 are currently pending in the application. Reconsideration of the present case is earnestly requested in light of the following remarks.

Telephone Interviews with Examiner

Telephone conversations were held between Examiner Sax and Mark S. Williams on June 17 and June 20, in which the above amendments were proposed to be made via Examiner's amendment to clarify the claimed invention. Distinctions between text-based programs / programming, and graphical programs / graphical programming were presented to the Examiner, as well as between manual creation of graphical programs and programmatic or automatic generation of graphical programs. Distinctions were also presented to the Examiner between illustrative diagrams and executable graphical programs. The Examiner agreed that these distinctions and proposed amendments were germane to the case and helped move the case forward.

Section 103 Rejections

Claims 1, 2, 5-33, 36-47, 50-59, 62-65, 68-76, and 79-81 were finally rejected under 35 U.S.C. 103(a) as being unpatentable over Volk et al (5673401, "Volk") and Morganelli et al (6425120, "Morganelli") and Poirier et al (6321372, "Poirier"). Applicant respectfully disagrees.

Amended claim 1 recites:

1. A computer-implemented method for programmatically modifying a graphical program, the method comprising:

executing a graphical program generation (GPG) program;

the GPG program receiving information, wherein the information specifies desired functionality of the graphical program;

the GPG program programmatically modifying the graphical program in response to said information specifying the desired functionality of the graphical program, such that the graphical program implements the specified desired functionality;

wherein the graphical program comprises a flow diagram comprising a plurality of interconnected nodes that visually indicate the functionality of the graphical program, wherein the graphical program is executable to perform said functionality according to the flow diagram;

wherein said programmatically modifying the graphical program is performed without any user input specifying the modification.

Applicant respectfully submits that there are numerous limitations of amended claim 1 not taught by Volk, Morganelli, and Poirier, as will be explained in detail below.

Applicant notes that in the previous Response, which is hereby incorporated by reference in its entirety, Applicant requested clarification regarding several issues in the previous Office Action. Applicant submits that the Examiner has not clarified these issues, and has continued to rely on these same ambiguous (to Applicant) terms and phrases in the current arguments without clarification. For example, regarding Volk, Applicant requested clarification as to the meaning of the Examiner's phrase "the underlying program generation" and notes that this terminology is not used in claim 1. As another example, with reference to both Volk and Morganelli, Applicant requested clarification regarding the phrase "without the user being involved in the inner processing of the software", which is also not used in claim 1. Similarly, with respect to Morganelli, Applicant requested clarification regarding the Examiner's use of the phrase "the program generation program aspect" and notes that this terminology is not used in claim 1 either. Applicant respectfully requests that the Examiner provide more detail regarding the assertions and their support in the referenced art.

Applicant further notes that in the previous Response, Applicant provided specific arguments against numerous of the Examiner's assertions in the previous Office Action, specifically addressing the citations used to support the assertions. However, in the current Office Action, the Examiner has simply stated that the arguments have been considered but are moot in view of the new grounds of rejection, yet has provided the

same assertions with the same citations in the current Office Action, and has not addressed Applicant's arguments. While the Examiner has provided further arguments and assertions based on the new reference (Poirier), Applicant respectfully submits that Applicant's arguments have apparently not been considered, or, if they have been considered by the Examiner, have not been addressed in the most recent Office Action. Applicant respectfully requests that the Examiner withdraw finality in the case and respond properly to Applicant's arguments.

Regarding Volk, the Examiner asserts (again) that Volk shows "the method for modifying a graphical program including executing a graphical program", citing Figure 1, 16A-B, and column 5, lines 40-55.

As argued previously, Volk neither teaches nor suggests graphical programs, nor a graphical program generation (GPG) program programmatically modifying the graphical program in response to information specifying the desired functionality of the graphical program, where the modifying is performed without any user input specifying the modification during the modifying.

Applicant notes (as discussed with the Examiner by telephone) that graphical programs are themselves executable programs, and are not simply diagrams illustrating functionality of some other programs. At the suggestion of the Examiner, Applicant has amended the independent claims above to emphasize that the graphical program is executable to perform the program functionality, i.e., that the graphical program is itself an executable program that executes to perform the functionality visually represented by the graphical program. In other words, 1) the graphical program is a program, not simply an illustrative diagram of some specified functionality, and 2) the graphical program is executable to perform the visually represented functionality.

Applicant further notes (as also discussed with the Examiner by telephone) that as used in the present application, the term "programmatic" means "automatic", i.e., *not performed manually*. The Examiner and Applicant agreed to amend the independent claims as indicated above to clarify that the programmatic modification of the graphical program is performed without any user input specifying the modification. In other

words, the modification is made to graphical program without any user input editing the graphical program.

Applicant notes that in contradistinction to claim 1, Volk teaches *manually* creating/programming a GUI. More specifically, Volk is directed to a system for generating and displaying control items that allow users of an interactive network to recognize and select control functions via a graphical user interface (Abstract). Regarding the Examiner's citations of Volk in the Office Action, which describe the display of a graphical user interface that utilizes control items, col. 5 lines 30 – 45 describes control items such as buttons, spin dials, and other visual objects displayed on the graphical user interface; Figures 16A-B flowchart the use of these objects; col. 6 lines 9 – 17 describes the control items that include audiovisual elements; col. 6 lines 35 – 60 describes control items having inheritability features that simplify the programming of user interface effects; col. 10 lines 15 – 38 describes focus items and focus elements, e.g., a focus item is described as something that is perceptible to a user, such as a screen display or a sound track.

Thus, as argued previously, Volk relates generally to the display of a graphical user interface, e.g., in an interactive television-based system. Applicant submits that it is well known in the art that a graphical user interface is not at all the same as a graphical program. As recited in claim 1, "the graphical program comprises a flow diagram comprising a plurality of interconnected nodes that visually indicate functionality of the graphical program, wherein the graphical program is executable to perform said functionality according to the flow diagram". The concept of an executable flow diagram (graphical program) that includes a plurality of interconnected nodes that visually indicate functionality of the program is nowhere taught or suggested in Volk. Thus, as noted previously, Volk does not teach the concept of a graphical program at all, and therefore cannot possibly teach the concept of modifying a graphical program, either programmatically or otherwise.

The Examiner admits that Volk does not teach "program generation per se", but asserts that Volk teaches "effectively modifying and storing the changes without the user being involved in the inner processing of the software", citing col. 10, lines 8-28, Figures 5-6, and col. 22, lines 10-50.

Applicant does not understand what the Examiner means by "without being involved in the *inner processing* of the software", nor how this relates to the subject matter of claim 1. Applicant notes that "inner processing" is not mentioned in either the present application or the referenced art. Figures 5-6 illustrate control items in a graphical user interface. Furthermore, col. 10, lines 8-28 of Volk actually reads:

> For example, a focus object may operate to alter the view of the selected control item, or to associate a separate focus item with the selected control item. When a control item receives focus, a function member of the control object associated with the selected control item can be executed in response to an action command transmitted via an input device.
>
> As used herein, a "focus item" is an aspect of a focus object that is perceptible to a user, such as a screen display or a sound track. For example, in a "Video On Demand" context, control items embodied as visual "buttons" may be displayed to a viewer, wherein each button represents a video that the viewer may select for viewing. A focus item may also be displayed to highlight or otherwise emphasize a particular one of the buttons. This supplies an indication to the viewer that the particular video associated with the highlighted button will be chosen if the viewer communicates an action command. A focus item may be present as an aspect of a control item or as a separate item.

Applicant submits that this portion of Volk in no way describes or even hints at programmatic modification (by a GPG program) of a graphical program without user input specifying the modifying during the modifying. Similarly, col. 22, lines 10-50 read:

> FIG. 5 illustrates an exemplary spin dial 104 that may be presented by the graphical viewer interface 100. The spin dial 102 is a type of control item having various control functions, constructed with a number of subsidiary or child control objects and control items that are assembled and displayed on the display screen.
>
> Intuitively, the spin dial 104 represents the metaphor of an imaginary spinning dial (such as the spinning disk 160 shown in dotted lines) that includes a number of text elements 162a-162f. The text element 162a, "Pepperoni", is visible in a text box 164 positioned centrally of a box shape 161. A left arrow 166a and a right arrow 166b are positioned adjacent the text box 164 and serve as spin controls. An indicator bar 168 is positioned beneath the text box 164, and includes a list position indicator 170 that shows the

> relative position of the text item 162a with respect to a list of several text items, e.g., "Cheese" 162b, "Hamburger" 162c, etc.
>
> In execution, the user only sees the spin dial control item 104, but the illusion is created of the existence of an imaginary spinning dial represented by the dial 160. By placing the cursor 112 on the left arrow 166a or right arrow 166b and pressing the action button 126 (in the roam mode), or hitting the left-hand navigation key 124a or right-hand navigation key 124b (in either the roam mode or the tab mode), the imaginary disk 160 is caused to "spin", bringing another text element, such as the text element 162b or 162c, depending upon direction, into position in view in the text box 164. For example, if the text element 162a is "Pepperoni", hitting the arrow 166a causes the item "Hamburger" to rotate into view, while hitting the arrow 166c causes the item "Cheese" to rotate into view. As the text rotates into position, the list position indicator 170 slides along the indicator bar 168 to show the relative position of the new term relative to the remaining terms in the list.
>
> In accordance with the present invention, control items such as the spin dial 104 are constructed in an object-oriented framework by assembling various control elements and displaying them at run time, as a function of programming..

Applicant submits that this portion of Volk describes a graphical user interface and its use, specifically, control items (e.g., spin dial, navigation buttons) in a GUI presented on a television, and in no way describes a graphical program, nor a GPG modifying the graphical program without user input specifying the modifying during the modifying. Applicant notes that in fact, the last paragraph of this text specifically states that the GUI (which is not a graphical program) is constructed in an object-oriented framework as a function of programming. In other words, as described in this text and elsewhere in Volk, any modifications in Volk's programs (which are not graphical programs as defined and described in the present application) are made directly by user programming, i.e., at the time of the modifying.

As Applicant noted earlier, as used in the present application, the term "programmatically" means "automatically", i.e., via execution of a program. For example, page 8, lines 19-21 of the Specification recites:

> ...the GPG program may ***automatically, i.e., programmatically,*** add a portion of graphical program code implementing the specified functionality to the user's program. *(emphasis added)*

Nowhere does Volk teach or suggest programmatic or automatic generation of a program, and more specifically, Volk fails to teach or suggest programmatic or automatic generation of a *graphical program.*

Similarly, the Examiner asserts that Morganelli shows "the program generation program aspect, with underlying code programming for effectively modifying and storing the changes without the user being involved in the inner processing of the software", citing Figures 9, 11, 14A, col. 8, lines 40-60, and col. 22, lines 16-40.

Applicant again requests clarification of the Examiner's phrase "inner processing of the software".

Applicant (again) submits that Morganelli does *not* teach the concept of a graphical program generation (GPG) program operable to programmatically modify a graphical program without receiving any user input specifying the modification during said programmatically modifying. Rather, Morganelli teaches *manually* creating a flow diagram.

As noted previously, Morganelli's FIG. 9 shows a data/control flow diagram 900 presented in a development environment GUI. Applicant submits that FIG. 9 specifically does *not* show a graphical program generation program. Figure 11 shows another view or instance of this GUI, also showing a circular flow diagram. Nowhere does FIG. 11 illustrate a graphical program generation program. Figure 14A shows yet another view or instance of the GUI with a more complex flow diagram. Applicant notes that per col. 31, lines 42-55:

> Using the icons of designer toolbar 414, ***the developer preferably creates a data/control flow diagram*** within the designer window 406 of the GUI 400 for performing such steps.
>
> FIGS. 14A-D are preferred illustrations of a GUI 1400 similar to GUI 400 described above showing some of the steps followed in creating such an application program. More specifically, within designer window 406 of GUI 1400 is a data/control flow diagram 1402 of the application program. The flow diagram 1402 includes a number of symbolic representations interconnected by a plurality

> of wire constructs. The symbols, moreover, correspond to respective program objects that have been instantiated and added to the form window 404 as described above. *(emphasis added)*

In other words, in Morganelli's system, the *developer* creates the flow diagram, i.e., *manually*.

Col. 22 lines 16 – 40 describe various objects in the data/control flow diagram 900 and their relationships to each other. The data/control flow diagram 900 apparently implements procedures operable to acquire a temperature measurement that is then utilized to make a comparison (see Col. 22 line 62 – Col. 23 line 19). However, as indicated in the previous Response, it is not at all clear what relevance this may have to the rejection of amended claim 1.

It is also unclear from the Office Action what role the Morganelli patent plays in the rejection of claim 1. In particular, it is not clear what specific elements recited in claim 1 the Examiner believes are not taught in Volk and what elements are believed to be taught in Morganelli. The Examiner states that, "Volk et al do not specifically show the underlying program generation per se". Applicant does not understand what the Examiner means by "the underlying program generation" and notes that this terminology is not used in claim 1. The Examiner goes on to state that, "Morganelli et al show the program generation program aspect". Again, Applicant does not understand what the Examiner means by "the program generation program aspect" and notes that this terminology is not used in claim 1 either. Applicant simply notes that Morganelli does not teach the concept of a graphical program generation (GPG) program operable to programmatically modify a graphical program without receiving any user input specifying the modification.

Regarding Poirier, Applicant notes (as discussed with the Examiner by telephone) that Poirier teaches the use of standard text-based object-oriented programming techniques in extending functionality to an existing code base. More specifically, Poirier teaches *manually* modifying pre-existing text-based code of an ancestor linguistic service to support new linguistic services.

The Examiner asserts that Poirier teaches "programmatically modifying a program without user specification of the modification during the actual modification", citing Figures 4, 6, 12, and 14, as well as col. 9, lines 30-65, col. 12, lines 50-65, and col. 18, lines 30-60, and asserting that "the user need not be involved with the inner processing of the software". Again, Applicant does not know what the Examiner means by "inner processing of the software", and respectfully requests clarification of this point. Applicant has read the cited portions of Poirier closely, and submits that Poirier nowhere teaches or suggests programmatically modifying a program without user specification of the modification.

In fact, Applicant submits that the cited figures and text of Poirier clearly indicate that the developer creates the additional functionality of the extended linguistic services by manually programming derivative program objects or classes based on the base classes, e.g., via inheritance, as is well known in the art of object-oriented programming. For example, Figure 4 is a schematic diagram showing an architecture for an environment in which the method of Poirier is performed, where the method itself is flowcharted in Figure 2. Applicant directs the Examiner's attention to step 62 of Figure 2 which recites: "MODIFY PREEXISTING SOURCE CODE TO OBTAIN MODIFIED SOURCE CODE FOR RESPONDING TO REQUESTS FOR NEW LINGUISTIC SERVICE". Applicant notes that in col. 9, lines 61-63, Poirier states:

> Then, in box 62, ***a human modifies the preexisting source code to obtain modified source code*** for responding to requests for a new linguistic service. *(emphasis added)*

Clearly, the source code is *not* modified programmatically or automatically, but rather via standard object-oriented *manual* programming techniques. Applicant further notes that Poirier nowhere teaches or suggests, or even mentions or hints at graphical programming, nor automatic or programmatic modification of graphical programs.

Thus, for at least the reasons provided above, Applicant respectfully submits that the cited references, taken singly or in combination, do not teach all of the elements, much less the combination of elements, recited in claim 1. Applicant thus submits that

independent claim 1, and claims dependent thereon, are patentably distinct over the cited references, and are thus allowable.

Inasmuch as independent claims 29, 30, 44, 45, 46, 56, 65, and 73 contain similar limitations as claim 1, Applicant submits that these claims, and claims respectively dependent thereon, are also allowable for reasons similar to those provided above.

Removal of the 103 rejection of claims 1-2, 5-33, 36-47, 50-59, 62-65, 68-76, and 79-81 is respectfully requested.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-52300/JCH.

Also enclosed herewith are the following items:

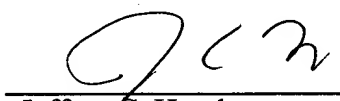
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Check in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,



Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 6/27/2005 JCH/MSW